Lecture 33: The Big 3 in C++

CS 62
Spring 2013
Kim Bruce & Kevin Coogan

Classes & Pointers in C++

- Book talks about "Big Three" for class D:
  - destructor: ~D() // cleans up when object deallocated
    - Could be deallocated explicitly by "delete" or implicitly by being local variable whose activation record is popped off run-time stack
  - copy constructor: D(D& other) // create new D like old
    - Used in initialization:
      D mycopy = existingD
      D mycopy(existingD)
      equivalent statements
  - operator= // used in assignments.
    - D fst, snd; //initialized to default values
      fst = new D(...);
      snd = fst;
      see which uses operator=

Big 3

- Not required for today's linked list lab
- Definitely required for Animals game!

Big 3

- Look at files in object_default and object_impl
  - Second group sets out explicitly what C++ does implicitly in the original.
  - Default copy constructor does shallow copy of original - just copies over instance (member) variables.
  - For operator=, check for alias before copying.
  - Destructors in example not do anything since no pointers in instance vbles.

First Attempt w/Pointers

- Ptr_shallow_impl
  - Converted instance vbles to pointers.
  - Now use new to initialize
  - Destructors use delete to explicitly deallocate
  - Shallow copy shares too much, code breaks!
    - Problem with copy constructor and operator=.
    - Worse operator= generates memory leak when assign to b3 old values of instance vbles lost.
    - Crashes when deallocators run because shared instance variables deleted multiple times.

For simplicity, put all code in .h files!

Better

- In ptr_impl
  - Make deep copies with copy constructor and operator=
- Your classes should look like this!!

Miscellaneous

- Like Java, in C++ if no constructor, will insert default parameterless constructor that just allocates space -- no initialization!
- If want to disable default copy constructor or operation=, just move to private section.
- Copy constructor should always involve "new"
- Type of "this" in class definition of D is D* const

Protection

- In Java: public, protected, private, & "default"
- In C++: public, private, and *friends*
- *Friends get to see your "privates"*

Example

```
class ListNode {
  public: ...
  private:
     int element;
     ListNode *next;
     ListNode(int theElement, ListNode *n)
              : element(theElement), next(n) {}
  friend class IntQueue;
```

If only want one method to have access, write:

```
  friend void IntQueue::enqueue(int x)
```